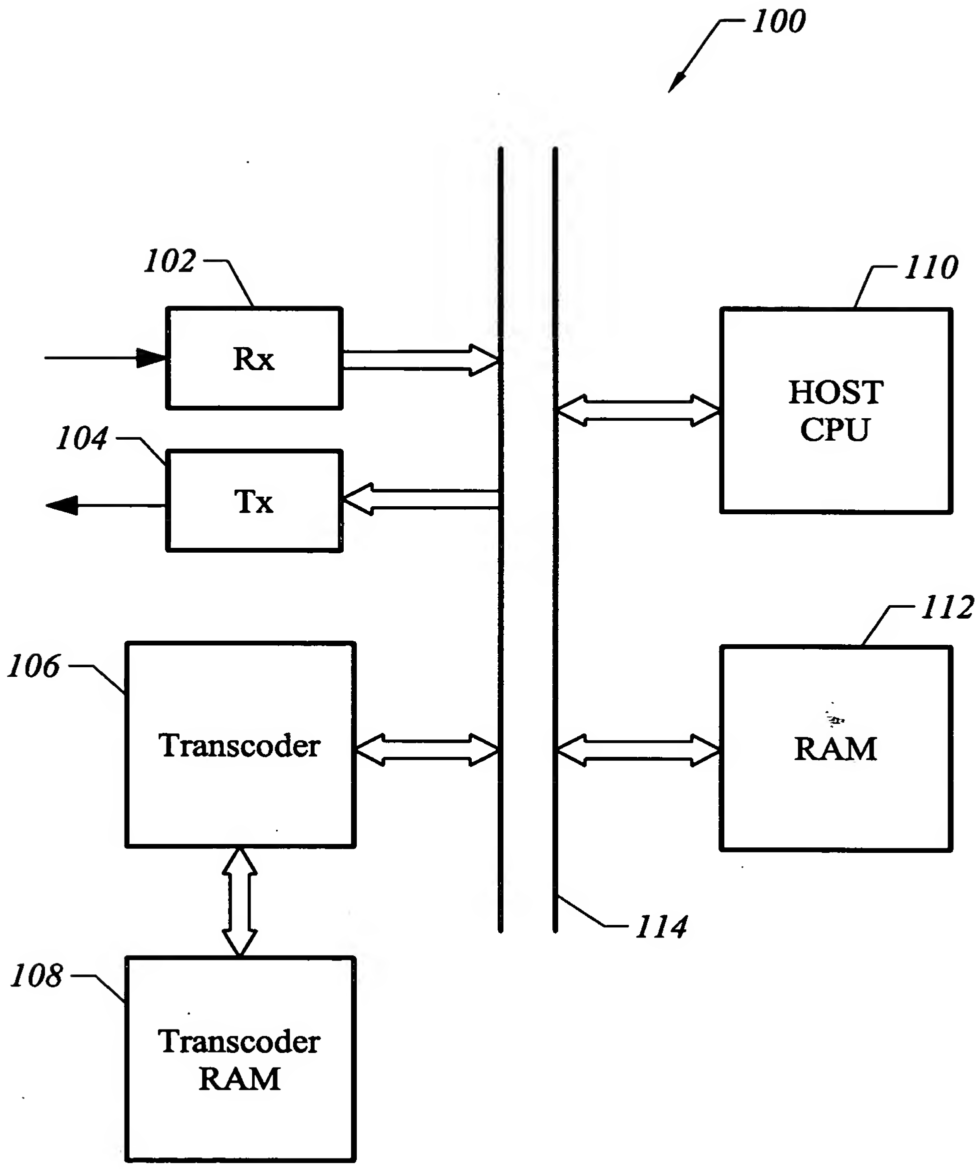

100
102
Rx
104
Tx
106
Transcoder
108
Transcoder RAM
110
HOST CPU
112
RAM
114

FIG. 1

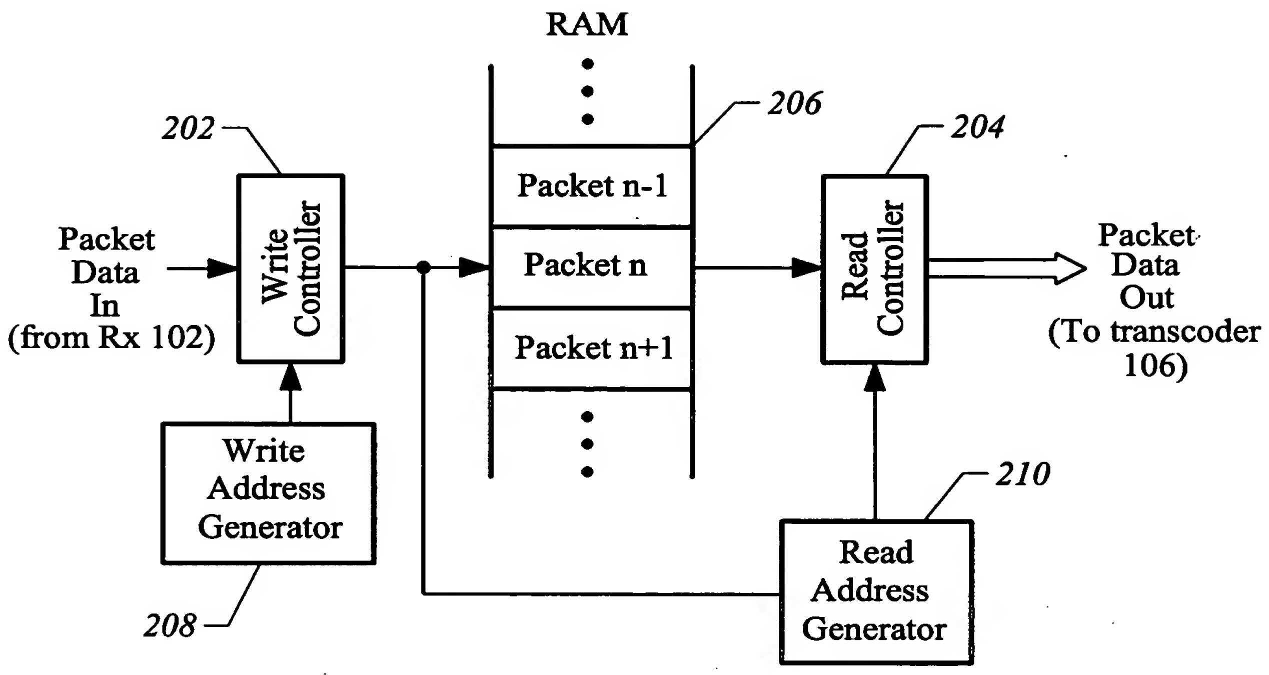

RAM
206
202
204
Packet n-1
Packet n
Packet n+1
Packet Data In (from Rx 102)
Write Controller
Read Controller
Packet Data Out (To transcoder 106)
Write Address Generator
208
210
Read Address Generator

FIG. 2

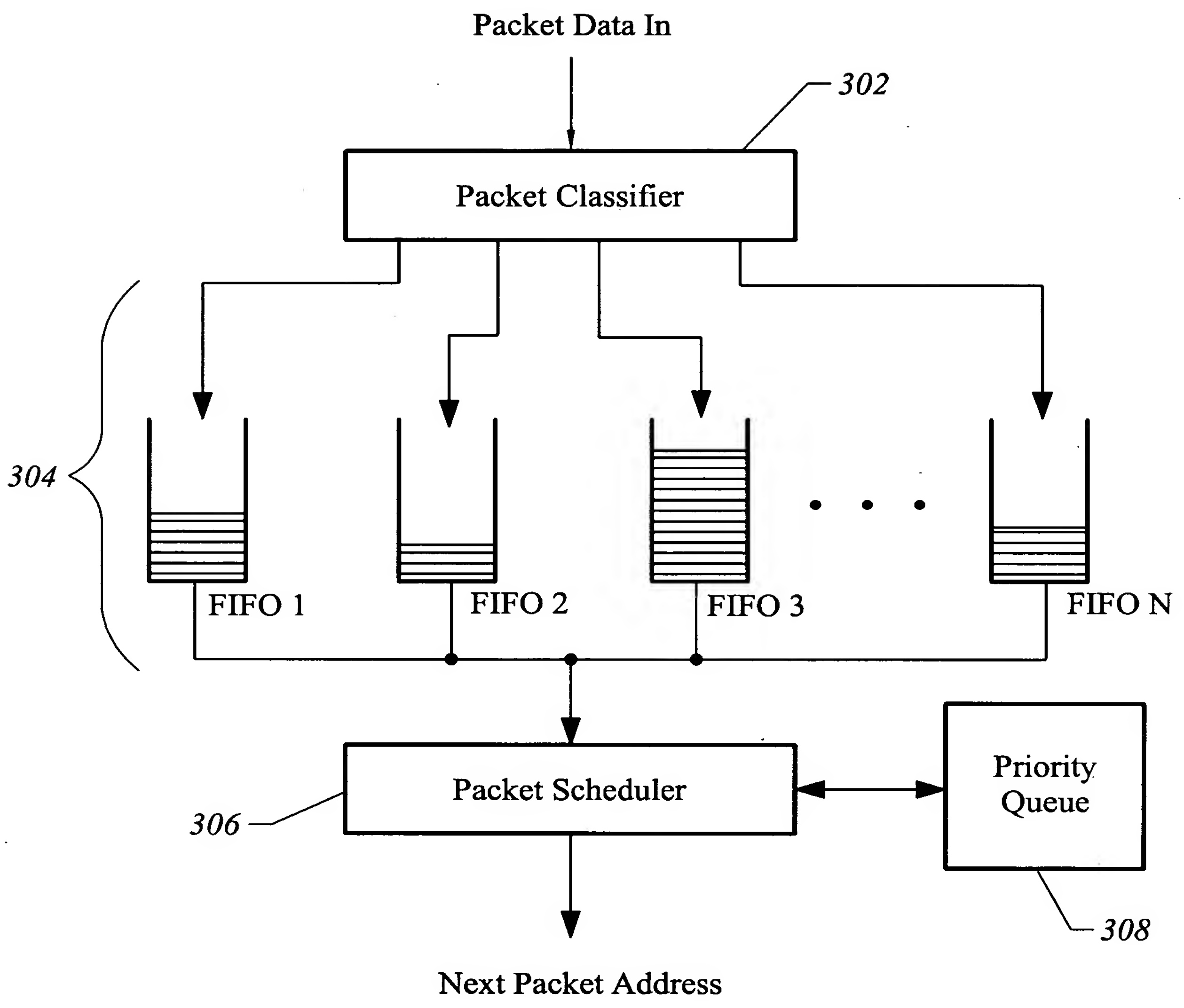

Packet Data In
302
Packet Classifier
304
FIFO 1
FIFO 2
FIFO 3
FIFO N
Packet Scheduler
306
Priority Queue
308
Next Packet Address

FIG. 3

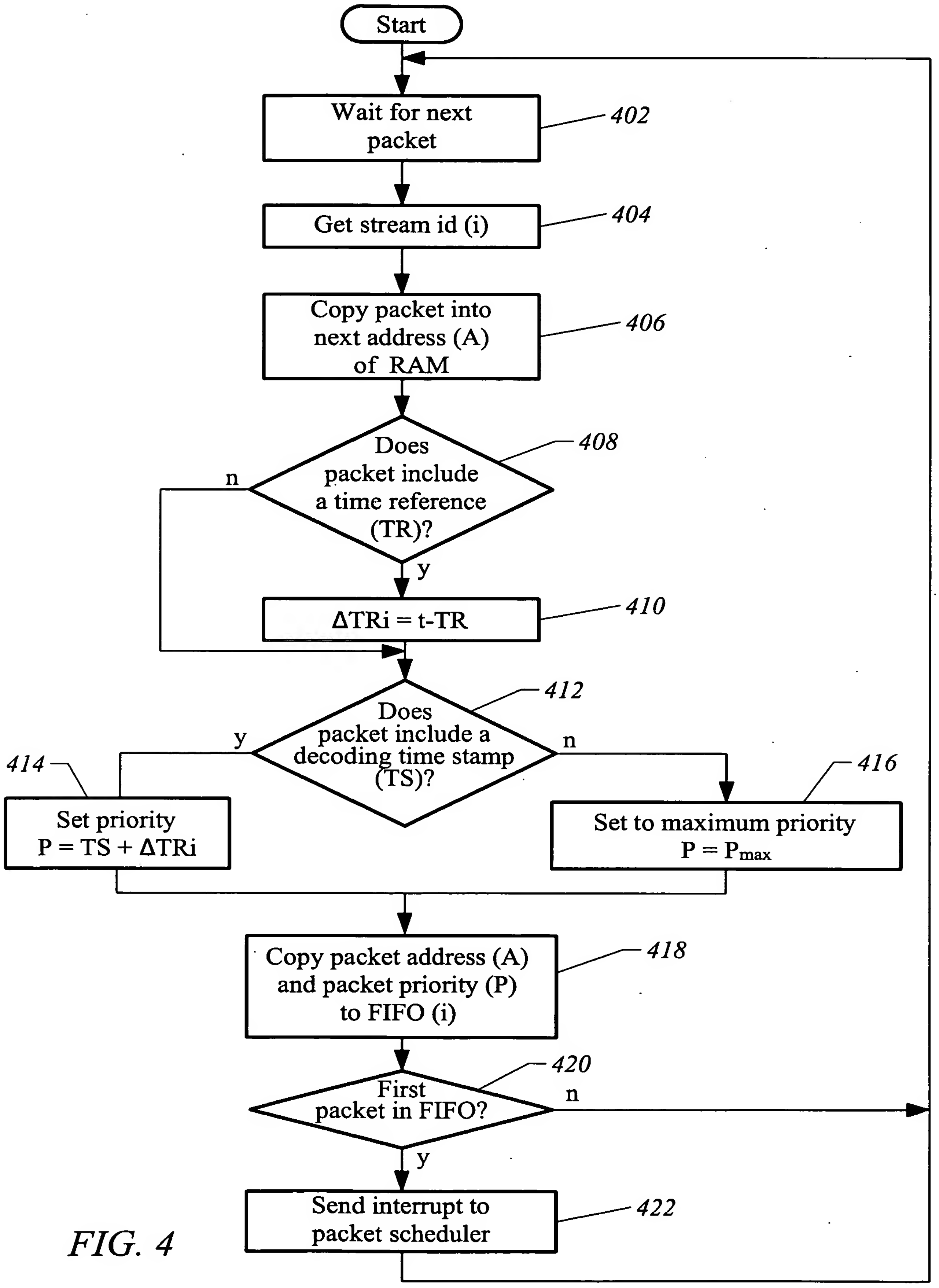

Start
Wait for next packet
402
Get stream id (i)
404
Copy packet into next address (A) of RAM
406
Does packet include a time reference (TR)?
408
n
y
ΔTRi = t-TR
410
Does packet include a decoding time stamp (TS)?
412
y
n
414
Set priority P = TS + ΔTRi
Set to maximum priority P = Pmax
416
Copy packet address (A) and packet priority (P) to FIFO (i)
418
First packet in FIFO?
420
n
y
Send interrupt to packet scheduler
422

FIG. 4

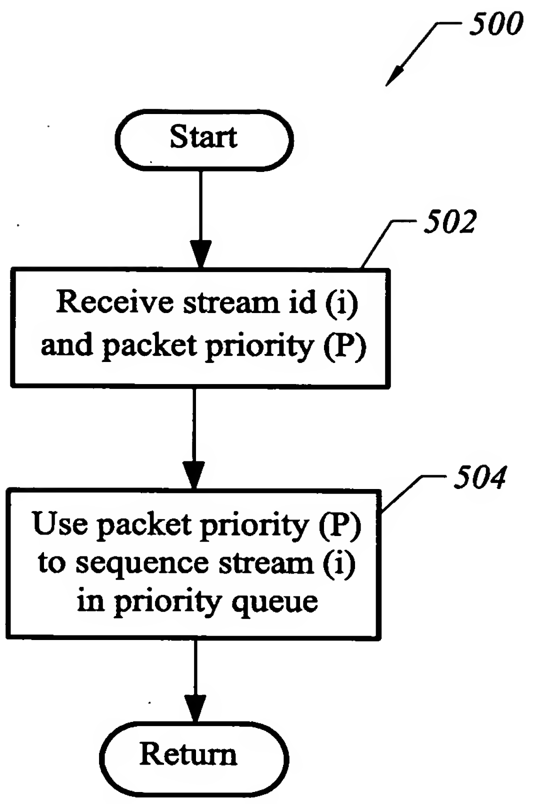
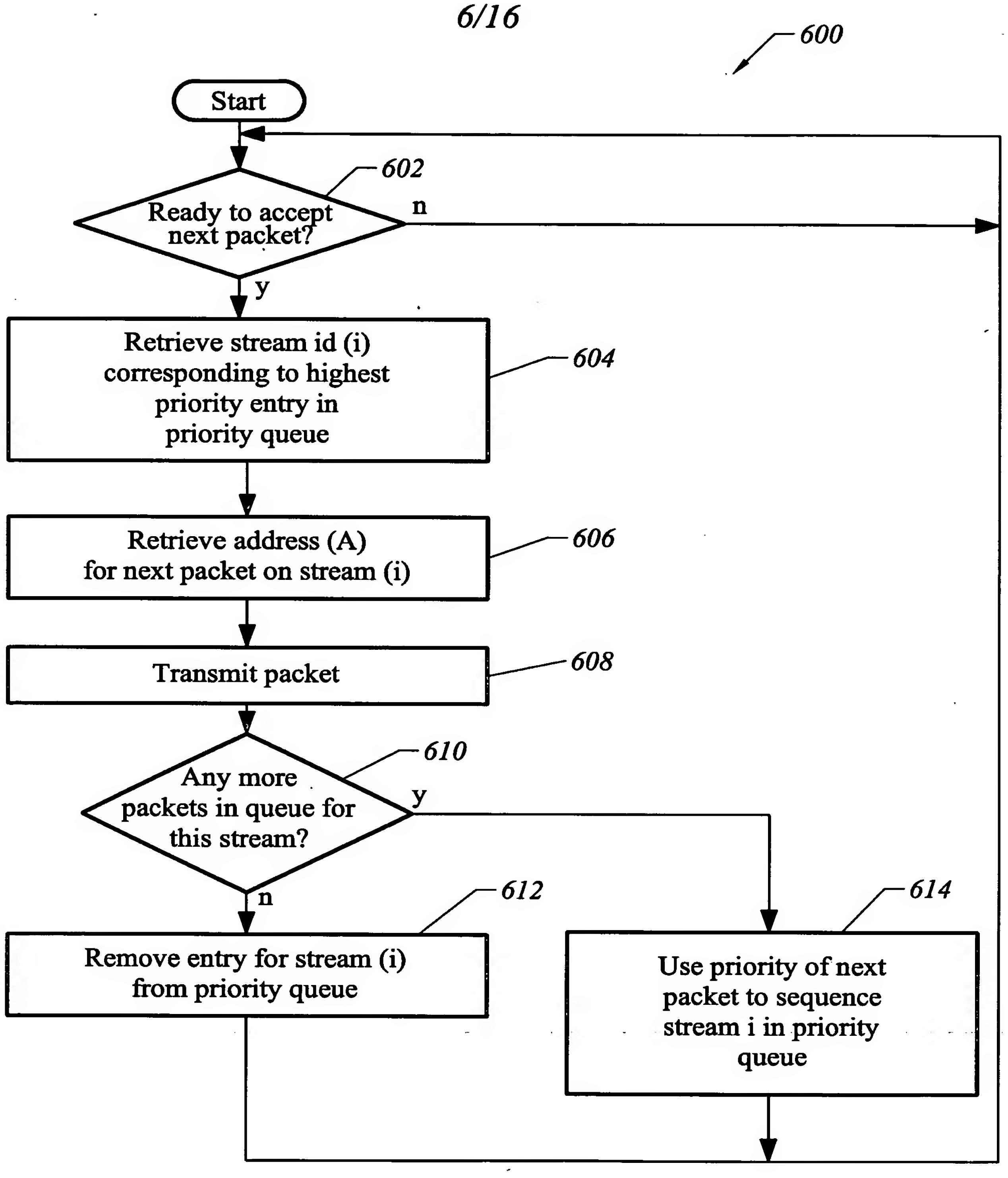

500
Start
502
Receive stream id (i) and packet priority (P)
504
Use packet priority (P) to sequence stream (i) in priority queue
Return

FIG. 5



600
Start
602
Ready to accept next packet?
n
y
Retrieve stream id (i) corresponding to highest priority entry in priority queue
604
Retrieve address (A) for next packet on stream (i)
606
Transmit packet
608
Any more packets in queue for this stream?
610
y
n
612
Remove entry for stream (i) from priority queue
614
Use priority of next packet to sequence stream i in priority queue

FIG. 6

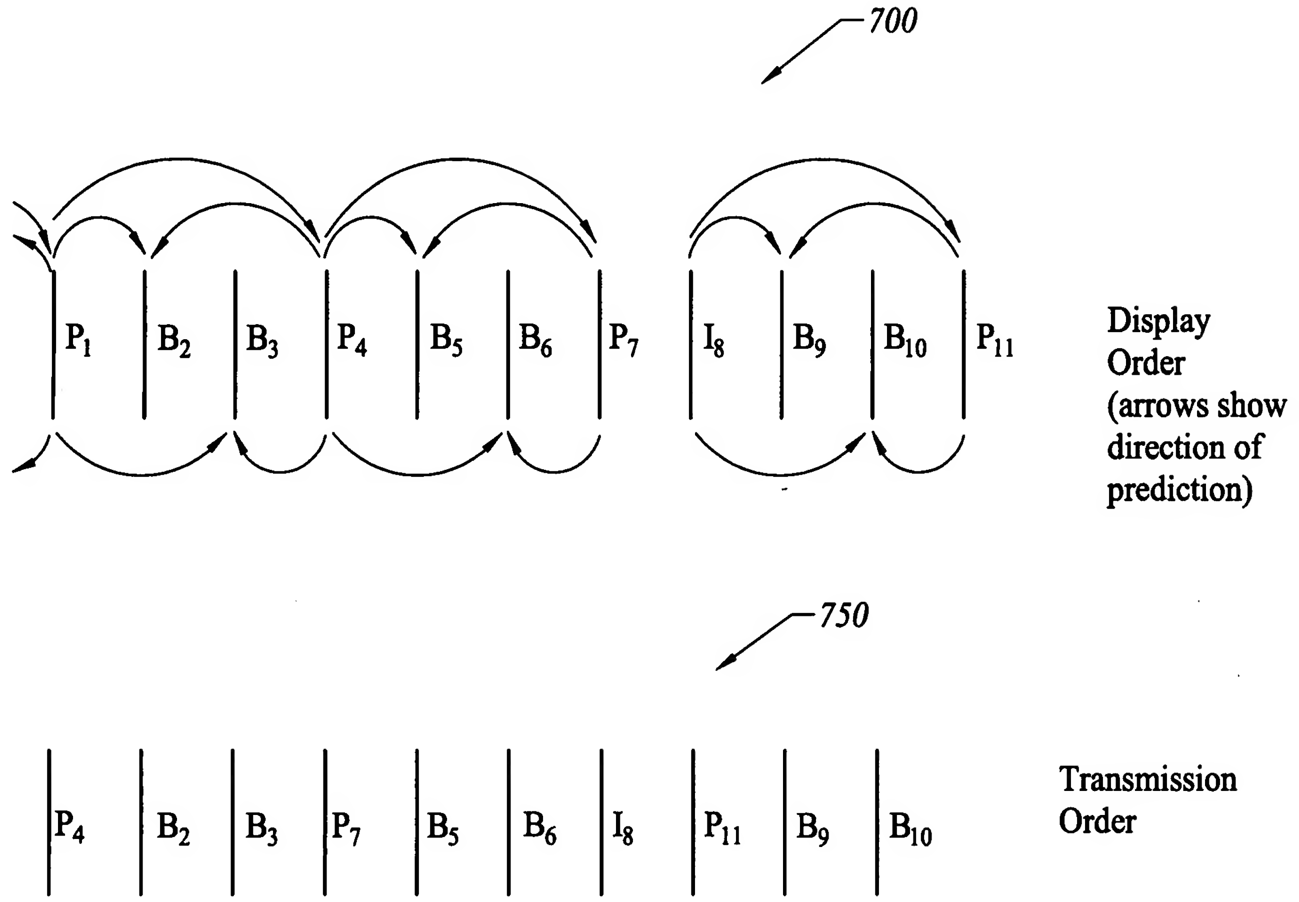

700
P_1
B_2
B_3
P_4
B_5
B_6
P_7
I_8
B_9
B_{10}
P_{11}
Display Order (arrows show direction of prediction)
750
P_4
B_2
B_3
P_7
B_5
B_6
I_8
P_{11}
B_9
B_{10}
Transmission Order

FIG. 7

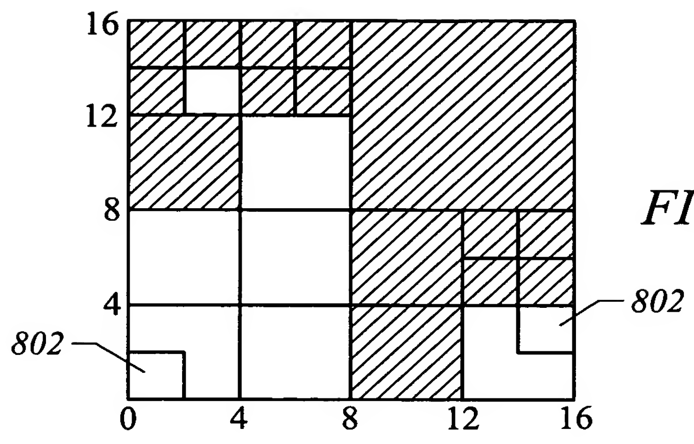


FIG. 8A

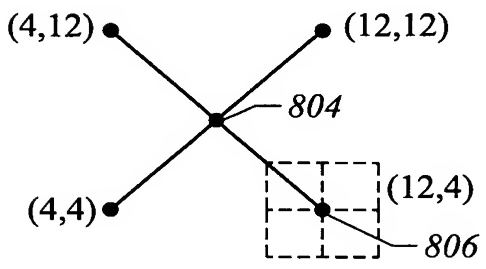


FIG. 8B

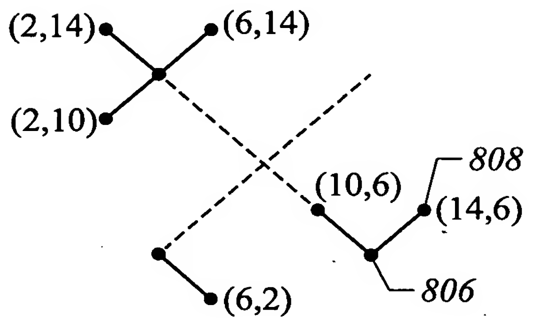


FIG. 8C

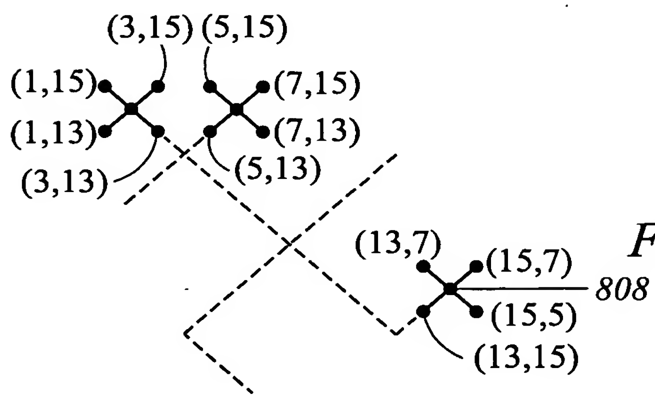


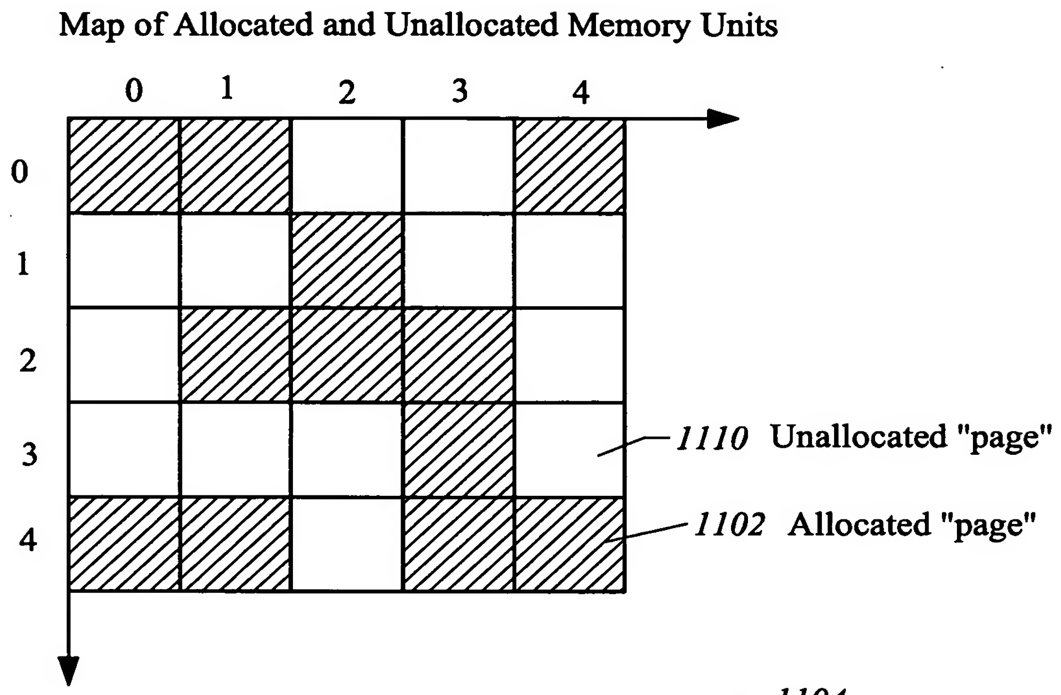
FIG. 8D

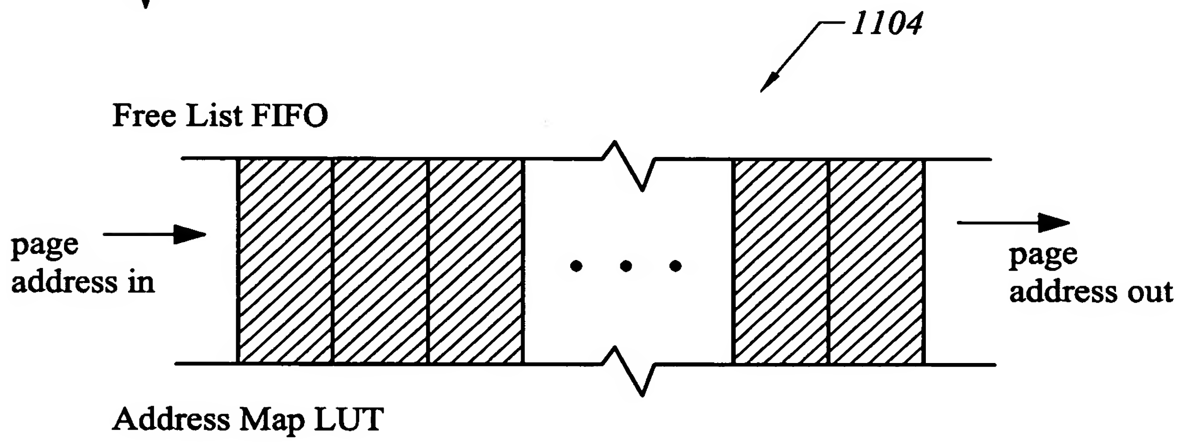
```
mem_allocate (d, i, j, k) begin
  if (d > k) begin
     D(i, j) = 0
     return (addr(i, j))
end
k=k / 2
if (d <= D(i+k, j+k) and
    (d > D(i+k, j-k) or D(i+k, j-k) >= D(i+k, j+k)) and
    (d > D(i-k, j+k) or D(i-k, j+k) >= D(i+k, j+k)) and
    (d > D(i-k, j-k) or D(i-k, j-k) >= D(i+k, j+k)))
     a = mem_allocate( d, i+k, j+k, k)
else if (d <= D(i+k, j-k) and
           (d > D(i-k, j+k) or D(i-k, j+k) >= D(i+k, j-k)) and
           (d > D(i-k, j-k) or D(i-k, j-k) >= D(i+k, j-k)))
     a = mem_allocate ( d, i+k, j-k, k)
else if ( d <=D(i-k, j+k) and
           (d > D(i-k, j-k) or D(i-k, j-k) >= D(i-k, j+k)))
     a = mem_allocate( d, i-k, j+k, k)
else
     a = mem_allocate( d, i-k, j-k, k)
D(i, j) = max( D(i+k, j+k), D(i+k, j-k), D(i-k, j+k), D(i-k, j-k))
return ( a )
end
```

FIG. 9

```
mem_free ( i, j, k ) begin
    D(i, j) = 2 * k
    while ( k < MEMSIZE/2 ) begin
       k = k * 2
       D(i, j) = max( D(i+k, j+k), D(i+k, j-k), D(i-k, j+k), D(i-k, j-k))
   end
end
```

FIG. 10


Map of Allocated and Unallocated Memory Units
0
1
2
3
4
0
1
2
3
4
1110 Unallocated "page"
1102 Allocated "page"


1104
Free List FIFO
page address in
page address out
Address Map LUT

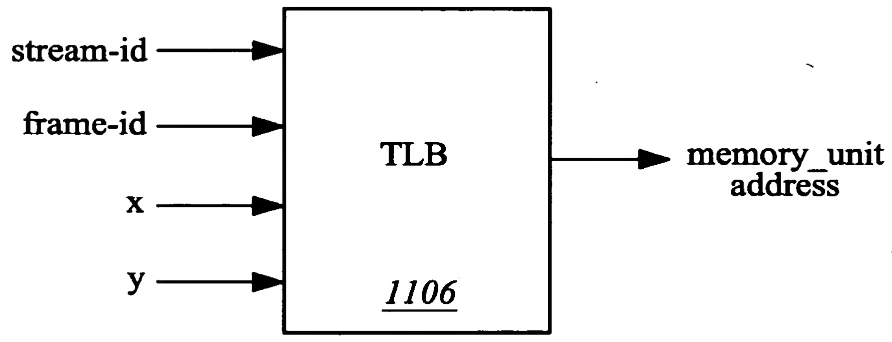

stream-id
frame-id
x
y
TLB
1106
memory_unit address

FIG. 11

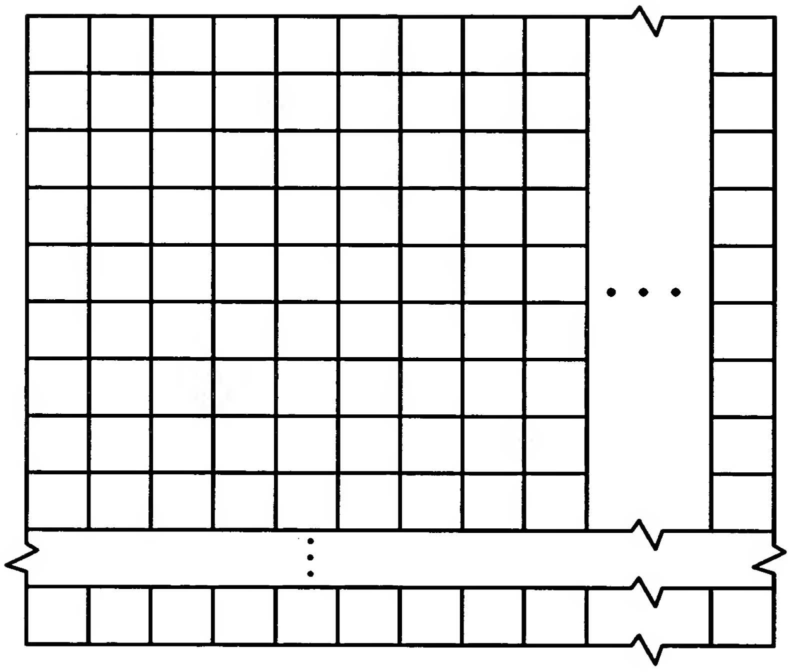


FIG. 12

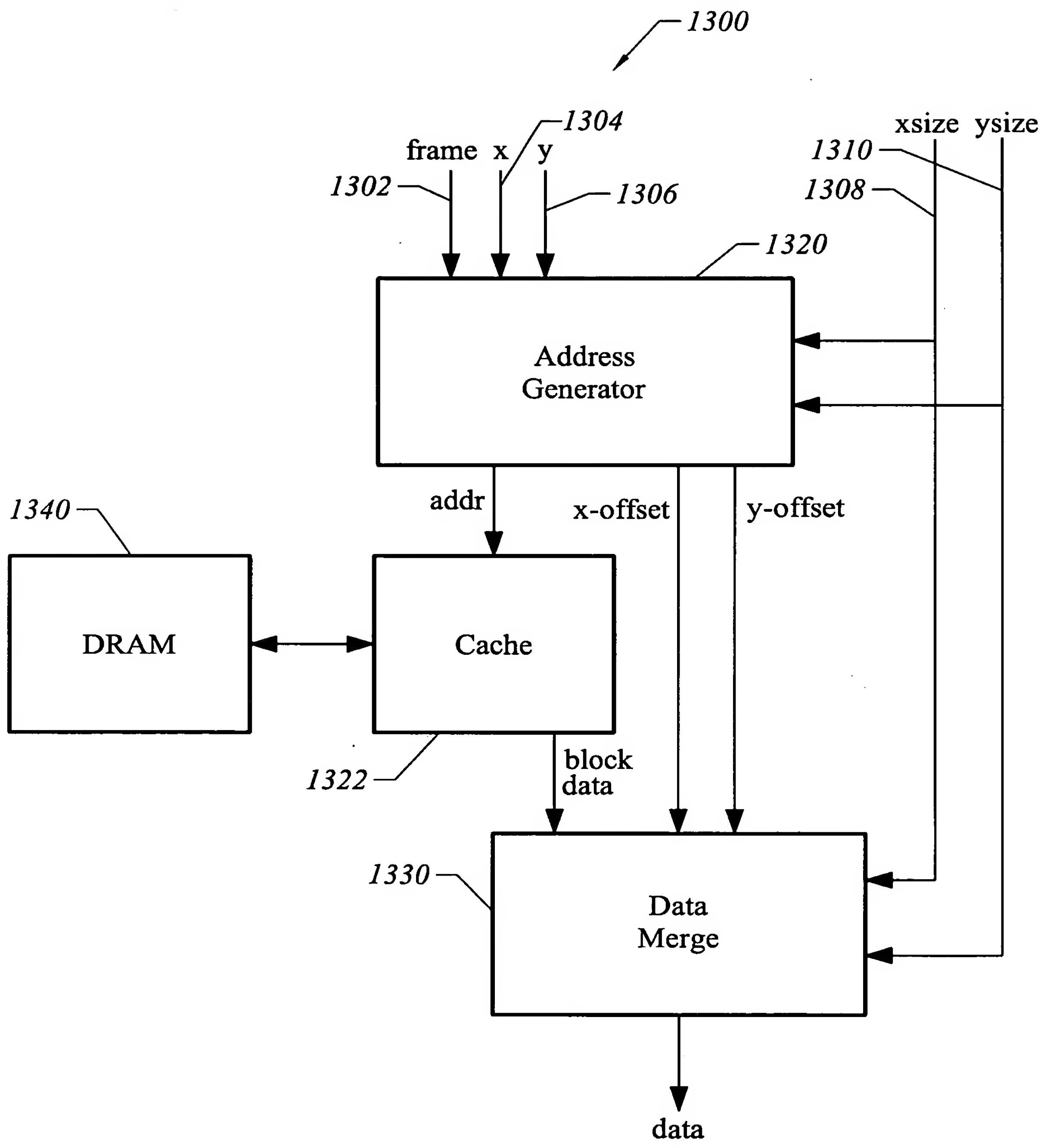

1300
1304
frame
x
y
1302
1306
1310
1308
xsize
ysize
1320
Address Generator
addr
x-offset
y-offset
1340
DRAM
Cache
1322
block data
1330
Data Merge
data

FIG. 13

```
address_generator ( ) begin
  m = 0
  n = 0
  input ( frame, x, y, xsize, ysize)
  while (n < ysize) begin
    x = xaddr + m
    y = yaddr + n
    block_addr = LUT { frame, y[ :7 ], x[ :7 ] }
    y_suboffset = y[6:4]
    x_suboffset = x[6:4]
    addr = { block_addr, y_suboffset, x_suboffset }
    y_offset = y[3:0]
    x_offset = x[3:0]
    output (addr, y_offset, x_offset)
    m = m + 16
    if ( m >= xsize ) begin
      n = n + 16
      m = 0
    end
  end
  return
end

data_merge ( ) begin
  input ( x_size, y_size, x_offset, y_offset)
  n = 0
  while ( n < y_size ) begin
    i = 0
```

FIG. 14

```
        while ( i < 16 ) begin
          m = 0
          while ( m < (x_offset + x_size) ) begin
            j = 0
            while (j < 16) begin
              input ( block_data )
              B[i][m] = block_data
              m = m + 1
              j = j + 1
            end
          end
          i = i + 1
        end
        if ( y_offset  > 0 ) begin
          i = y_offset
          y_offset = 0
        else
          i = 0
        end

        while ( i < 16 and n < ysize) begin
          while ( j < x_size ) begin
            data = B[i][j + x_offset]
            output ( data )
            j = j + 1
          end
          i = i + 1
          n = n + 1
        end
      end
    end
```

FIG. 15

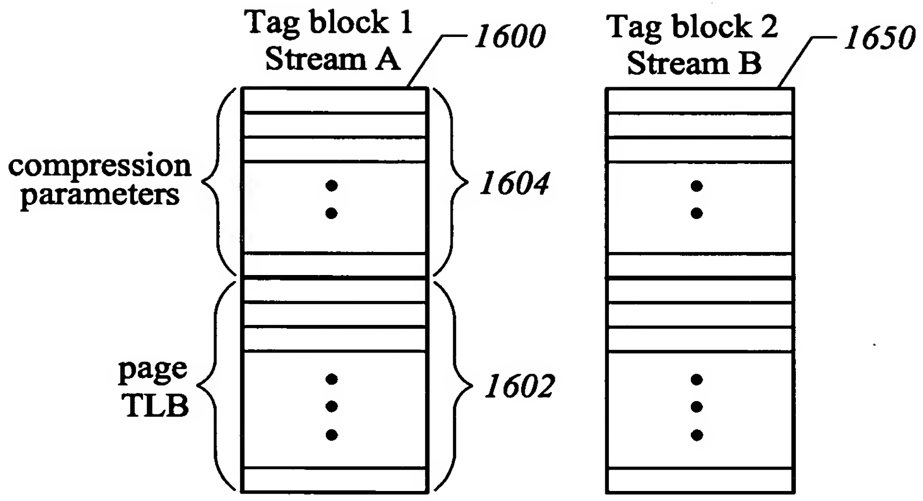

Tag block 1
Stream A
1600
Tag block 2
Stream B
1650
compression parameters
1604
page TLB
1602

FIG. 16

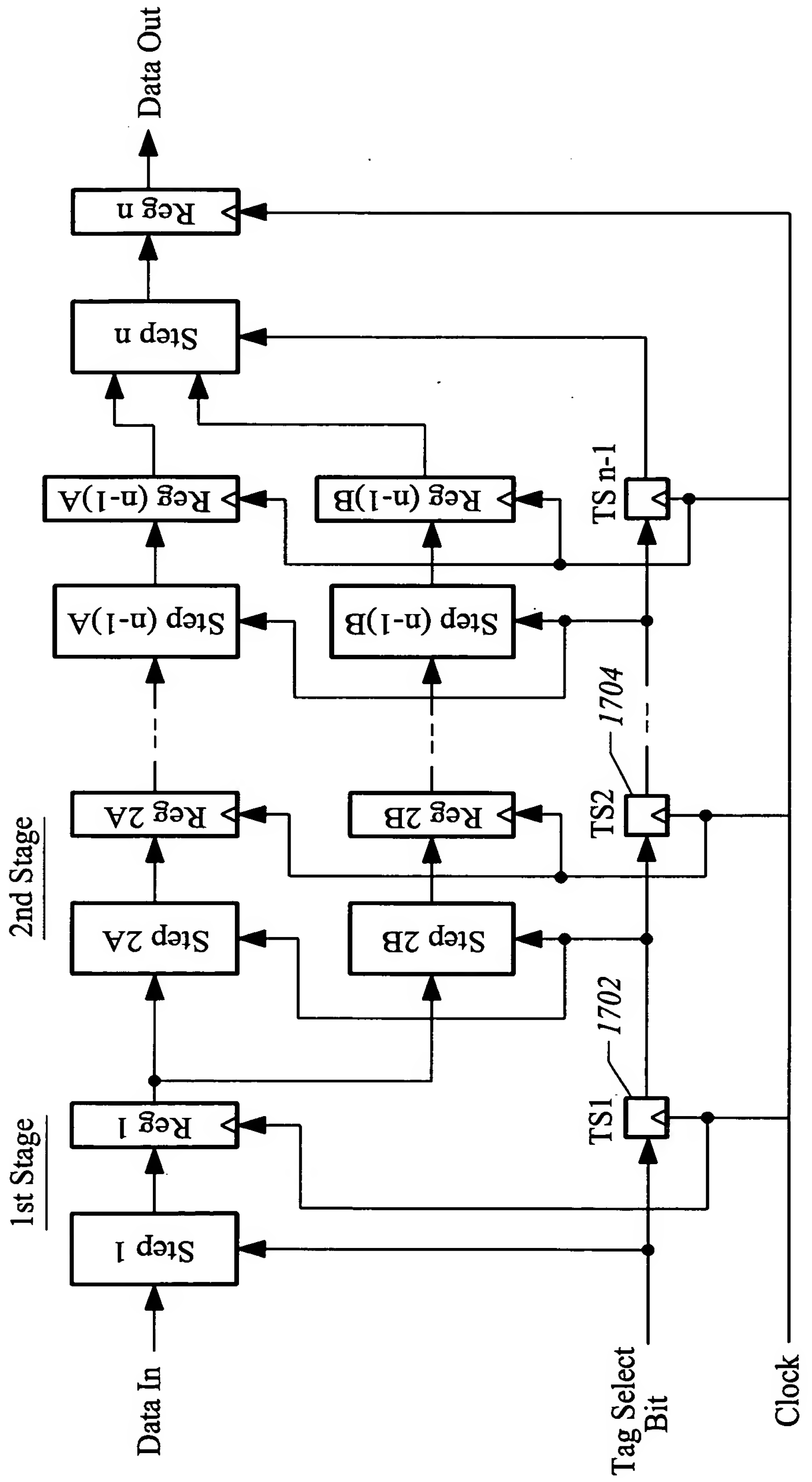

Data In
Step 1
Reg 1
1st Stage
Step 2A
Reg 2A
2nd Stage
Step 2B
Reg 2B
Step (n-1)A
Reg (n-1)A
Step (n-1)B
Reg (n-1)B
Step n
Reg n
Data Out
Tag Select Bit
TS1
1702
TS2
1704
TS n-1
Clock

FIG. 17